Docket No.: 42390P9874 *Utility Patent*

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re the Patent Application of:	)	
Khare et al.	)	
Serial No.: 09/752,576	)	Examiner: Ho, T.
Assignee: Intel Corporation	)	
Filed: December 29, 2000	)	
	)	Art Unit: 2188
For: MECHANISM FOR INITIATING AN IMPLICIT WRITE-BACK IN RESPONSE TO A READ OR SNOOP OF A MODIFIED CACHE LINE	)	

Mail Stop Appeal Briefs - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF UNDER 37 CFR §1.192
IN SUPPORT OF APPELLANTS' APPEAL
TO THE BOARD OF PATENT APPEALS AND INTERFERENCES

Sir:

Appellants hereby submit this Appeal Brief, in triplicate, in support of Appellants' Appeal from final rejection of the pending claims in the above-captioned case.

A Notice of Appeal was filed on April 26, 2004.

The fees set forth in 37 CFR §1.17(c) accompany this Appeal Brief.

An oral hearing is NOT desired.

Appellants respectfully request consideration of this Appeal by the

honorable Board of Patent Appeals and Interferences, and allowance of the claims of the subject application.

Please charge any fees and/or credit any overcharges to Deposit Account No. 50-0221.

I. REAL PARTY IN INTEREST

The subject application is assigned to Intel Corporation of 2200 Mission College Boulevard, Santa Clara, California 95052-8119.

II. RELATED APPEALS AND INTERFERENCES

To the best of Appellants' knowledge, there are no appeals or interferences related to the present appeal that will directly affect, be directly affected by, or have a bearing on the Board's decision.

III. STATUS OF THE CLAIMS

Claims 1-4, 6-18, and 20-21 are currently pending in the subject application. These claims were finally rejected in the Final Office Action mailed December 18, 2003. The Examiner confirmed the final rejection of these claims in an Advisory Action mailed March 3, 2004 (hereinafter "Advisory Action").

In the Final Office Action, the Examiner raised three grounds of rejection. First, the Examiner rejected claims 1-4, and 6-14 under 35 U.S.C. §102(b) as being anticipated by Hamaguchi et al. (U.S. Patent No. 5,737,568). Second, the Examiner rejected claims 9-11 under 35 U.S.C. § 103(a) as being rendered obvious by the combination of Hamaguchi et al. in view of Flynn et al. (U.S. Patent No. 5,222,224). Third, the Examiner rejected claims 15-18, and 20-21 under 35 USC §103(a) as being rendered obvious by Hamaguchi et al. Appellants respectfully traverse each of these grounds of rejection.

IV. STATUS OF AMENDMENTS

In response to the Final Office Action, Appellants filed an Amendment, After Final Rejection, Under Rule 116 (hereinafter, "Amendment") on February 18, 2004. In the Advisory Action, the Examiner indicated that the Amendment would be entered.

In response to the Final Office Action and the Advisory Action, a Notice of Appeal was timely filed on April 26, 2004. A copy of all claims on appeal is attached hereto as Appendix A.

V. SUMMARY OF THE INVENTION

Brief initial definitions of terms used throughout this application are given below to provide a common reference point. (Specification, page 8, lines 2-3.)

A home node is a node where the contents of a cache line are permanently stored. (Specification, page 8, line 4.)

A responding node is a node that has a copy of the contents of the cache line in question and is responding to a read request for the most recent copy of the contents. (Specification, page 8, lines 5-6.)

A requesting node is a node that initiates a read request for contents of a particular cache line or memory. (Specification, page 8, lines 7-8.)

An implicit write-back is a response by a responding node to a read or snoop request directed to a modified cache line that automatically updates the memory in question on the home node. (Specification, page 8, lines 9-11.)

Figure 1 illustrates an exemplary operating environment 100 according to one embodiment of the invention. In this example, multiple nodes 110 and 120 share memory through a cache based coherence system. The nodes supported are processor nodes 110 each having a local memory 130 and Input/Output (IO) nodes 120. The cache based coherence system is collectively designated the Scalability Port (SP). In node environments with more than two nodes the SP includes a System Node Controller (SNC) chip 140 in each of the processor nodes 110 and an IO Hub (IOH) 150 chip in each of the IO nodes 120. The IO node implements a cache, such as an L2 cache, so that it may participate in cache coherency. In addition to the SNC 140 and the IOH 150, the SP provides central control for its snoop architecture in a Scalability Port Switch (SPS) 160 that includes a snoop filter (SF) 170 to track the state of cache lines in all the caching nodes. The SNC 140 interfaces with the processor bus 180 and the memory 130 on the processor node 110 and communicates cache line information to the SPS 160 when the line is snooped for its current status. Similarly, the IOH interfaces with the IO Bus and communicates information to the SPS 160 when a line is snooped for its current status. (Specification, page 8, line 13 – page 9, line 3; FIG. 1.)

The SP used to exemplify embodiments of the invention supports various architectures. For instance, the processor nodes 110 could be based on either the IA32 or IA64 architecture. Unlike prior snoop based cache coherence architectures, the SP supports the full MESI (Modified, Exclusive, Shared and Invalid) protocol as uniquely implemented by both architectures, the IA 32

coherence protocol as well as the IA64 coherence protocol. One example of how these coherence protocols differ is when the cache line is in a Modified state when a read request is initiated. In the IA32 coherence protocol, once the read request is processed, the state of the cache line transitions from Modified to an Invalid state whereas in the IA64 coherence protocol, the cache line, once read, transitions from a Modified state to a Shared state. The support of multiple architectures allows for scalability and versatility in the future development of architectures and their corresponding protocols by allowing for the resident component of the SP, i.e, the SNC for the processor node and the IOH for the IO Node, to be implemented to handle the new architecture and its corresponding protocol without having to redesign the central snoop controller, the SPS. (Specification, page 9, lines 4 –18; FIG. 1.)

For simplicity, the following description utilizes several processing nodes because the IO nodes function in a similar manner with regard to embodiments of the invention. The Snoop Filter in the SPS is organized as a tag cache that keeps information about the state of each cache line and a bit vector indicating the presence of the cache line at the various caching nodes. An illustration of the information maintained in the Snoop Filter 200 is demonstrated abstractly in Figure 2. The contents of memory location AAAA 210, maintained exclusively on the Home Node 220, are copied and accessible in a cache 230 on the responding node 240. The responding node SNC (or IOH) 250 maintains a local presence vector 260 and status 270 for each cache line it utilizes, a snoop to the SNC of node 240 may result in the Snoop Filter's presence vector and status

being updated. The bit vector, called the presence vector, has one bit per caching node in the system. If a caching agent at any node has a copy of the cache line, the corresponding bit in the presence vector for that cache line is set. A cache line could be in the Invalid, Shared, or Exclusive state in the Snoop Filter. In this case, the home node's cache line is in a Shared state (S), while the requesting node's cache line is in an Invalid state (I) and the responding node's cache line was last known to be in an Exclusive state (E). According to the described embodiment, the cache line in the Snoop Filter will not indicate that a line is in a Modified state, because a read to a Modified line will result in the Modified line changing states in response to a snoop or read inquiry. (Specification, page 9, line 19 – page 10, line 13; FIG. 2.)

The Snoop Filter is inclusive in that it does not contain the cache data, but only tracks the tag and the state of caches at all the caching agents. It is possible to divide the Snoop Filter into multiple Scalability Port Switches or into multiple caches within one SPS to provide sufficient Snoop Filter throughput and capacity to meet the system scalability requirement. In such cases, different snoop Filters keep track of mutually exclusive sets of cache lines. A cache line is tracked at all times by only one Snoop Filter. (Specification, page 10, lines 14 - 20.)

The state of a cache line in the Snoop Filter is not always the same as the state in the caching agent's SNC. Because of the distributed nature of the system, the state transitions at the caching agents and at the Snoop Filter are not always synchronized. In fact, some of the state transitions at the caching agents

are not externally visible and therefore it is not possible to update the Snoop Filter with such transactions. For example, transitions from an Exclusive state to a Modified state may not be visible external to the caching agent. Although other ambiguous situations may exist, the usefulness of embodiments of the invention is illustrated by the scenario in Figure 2 when a cache line is in the Exclusive state at the Snoop Filter. In this case, the Snoop Filter is aware only that the caching agent, i.e. the responding node 240, has exclusive access to the cache line as indicated by the presence vector. However, the state of the cache line at the caching agent may have changed to any of the other MESI states (e.g., Modified, Shared or Invalid). If a read request is made to the SPS 290 for a cache line where ambiguity may exist, the SPS snoops the cache line, in this case the responding node's cache line, indicated by the presence vector to get its current state and most recent corresponding data if necessitated. (Specification, page 10, line 21 – page 11, line 12; FIG. 2.)

Figures 3, 4, and 5 illustrate what happens in the example illustrated in Figure 2 where an ambiguity between the SF and the cache agent exists. In this example, the requesting node 280 is the node making the read request for the most current updated contents of memory location AAAA. The home node 220 is the node where the data is stored for memory location AAAA 210 and the responding node 240 is the node that currently has a modified copy of the data for memory location AAAA 230. When the responding node 240 originally acquired its copy of the data for memory location AAAA 230, the Snoop Filter 200 indicated that the responding node 240 had a copy by asserting its presence

bit vector and additionally indicated that the responding node 240 was taking the copy in an Exclusive State 291. Once the Snoop Filter identifies that the data resides on the responding node, it need not monitor the activity at the responding node until another request is made. In this case, the responding node modified the data from X to X+A on the cache line and consequently its local cache line state changed to Modified 270. (Specification, page 11, line 13 – page 12, line 2; FIGS. 3, 4, and 5.)

Figure 3 demonstrates the sequence of events producing an implicit write-back response by the responding node. In step 310, the requesting node submits a read request for the contents associated with memory location AAAA. At step 320, the SPS 290 directs the read request to last known owner of the data, or the responding node. In step 330, the responding node receives the request and generates both an answer to the read request and an implicit write-back. According to one embodiment, both the answer and the implicit write-back are contained in a single response by the responding node. In Step 340, the SPS directs a write to the home node and an answer to the requesting node. The home node, in step 350, updates memory location AAAA with the modified contents and generates a completion response. In step 360, the SPS directs the completion response generated by the home node to the Requesting Node. When the completion response has been received the entire transaction is completed. (Specification, page 12, lines 3-14; FIG. 3.)

Figure 4 demonstrates how the SPS determines where to direct the read request. In step 410, the SPS receives a read request from the requesting node

for memory location AAAA. The SPS then checks the SF presence vector table, in step 420, to see which node has the last copy of the cache line concerning memory location AAAA. In step 430, the SPS identifies that the Responding Node has a copy of the data and that its last known state was Exclusive. However, because the Exclusive state is a known ambiguous state at the Snoop Filter level, the SPS, in step 440 snoops the Responding Node by directing the read request to the responding node to identify the current status of the cache line so that it may properly address the read request from the requesting node. (Specification, page 12, lines 15-23; FIG. 4.)

Figure 5 demonstrates how the responding node responds to the read request routed to it by the SPS from the requesting node. At step 510, the responding node receives the read request from the SPS. Because the Responding Node is aware that it is not the home node for this cache line and knows that the cache line state is now changing to either a Shared or Invalid state in reaction to the read request depending on whether it is implementing the IA32 or IA64 coherence protocol, it identifies what state it will transition to as well as information indicating to the SPS which, if any or both, of the requesting and home nodes need to receive updated data. At step 520, the responding node determines if its copy of the cache line AAAA has been modified. In step 530, an implicit write-back is initiated by the responding node in response to the read request by generating a command instructing the SPS to update the data at the home node. As indicated above, the implicit write-back and an answer to the read request may be communicated together in a single response. The answer

to the requesting node may or may not include the modified data depending on the type of request made by the requesting node. The implicit write-back is then performed by the SPS by causing the home node to update memory location AAAA with the modified data. By performing an implicit write-back, a foundation is laid for the requesting node to assert exclusive control of the cache line so it may perform local modifications. After the home node performs the data update, a completion response is provided to the SPS that in turn is provided to the requesting node as demonstrated previously in steps 340, 350 and 360. Advantageously, this novel implicit write-back mechanism eliminates the additional time and resource burden at the requesting node as the responding node provides the information to accomplish the data update at the home node and the SPS need not wait for a write request from the requesting node. (Specification, page 13, lines 1-24; FIGS. 3, 5.)

Referring back to figure 2, the dashed arrows demonstrate the steps as discussed above according to one embodiment of the present invention concerning a read request and the resulting implicit write-back. At arrow 251, a processor on the requesting node submits a snoop request (i.e., a Port Snoop Line Data [PSLD] request) for Memory AAAA that resides on the home node. The request is transmitted 252 through the SNC on the Requesting Node to the SPS responsible for managing memory location AAAA and the Home Node. The SPS, by utilizing the Snoop Filter, identifies that the cache line for Memory AAAA was last in an Exclusive state on the responding node by examining the cache line's presence vector and corresponding status. Knowing that the Exclusive

state is an ambiguous state, the SPS passes 253 the requesting node's PSLD request to the responding node. If the responding node is still in an Exclusive state, it will indicate its state and the SPS will tell the requesting node to retry later. If, on the other hand, the cache line is in a Modified state, the responding node sends 254 a Port Snoop Node Response (PSNR), an indication of what state the cache line at the dirty node is transitioning to so the snoop filter may update its presence vector, and a command informing the SPS to both provide the data to the requesting node and to perform an update by providing the data to the home node. This updating of the home node in response to a read or snoop request directed to a modified cache line is referred to as an implicit write-back. The SPS updates its Snoop Filter with the new status of the cache line at the responding node and substantially simultaneously provides 255 the modified data to the home node and provides 256 the response from the responding node that may include the modified data to the requesting node. When the Home Node has successfully updated Memory AAAA, it sends a completion response 257 to the SPS that routes the completion response in step 258 to the requesting node and completes the transaction. (Specification, page 14, lines 1-24; FIG. 2.)

In the Scalability Port exemplified, the response packet (e.g., the PSNR) contains information necessary to accomplish the implicit write-back. Figure 6 illustrates information that may be communicated during physical transfers according to one embodiment of the present invention. In this example, each 40 bits communicated is considered a physical unit (PHIT). Phit(0) of the response packet includes information concerning the type of response (Resp Type) that is

being communicated, the state of the cache line snooped and information concerning where the data is to be routed. Although various information is provided in the information bits such as destination node and data length, of significance for embodiments of the invention are the Route bits Phit[0].Info[17:16] 610 and the Response Type bits Phit[0].Info[21:18] 620. (Specification, page 15, line 1-10; FIG. 6.)

The Route bits indicate whether the information is to be routed to the home node or the requesting node or both. The least significant bit 612 in this field, Route [0] or Info [16], indicates whether the data is to be routed to the requesting node or not. If it is to be routed to the requesting node, then the bit is asserted. The most significant bit 611 in this field, Route [1] or Info [17], indicates whether the home memory needs to be updated or not. When Route [1] is asserted, an implicit writeback occurs and the home node memory is updated. (Specification, page 15, lines 11-17; FIG. 6.)

The most significant bits 613 of the Response Type bits indicate whether the response is a snoop response. If both RespType[3:2] bits are asserted, the response is a snoop response and the least significant bits 614 RespType[1:0] indicate the state of the cache line. If the response is a snoop response, two of the possible four states are Modified transitioning to Invalid (PSNRM) and Modified transitioning to Shared (PSNRMS). If an incoming request is a port snoop line data (PSLD) request, A PSNRM response "1110" indicates that the architecture of the responding node is implementing the IA32 coherence protocol while a PSNRMS response "1111" indicates that the architecture of the

responding node is implementing the IA64 coherence protocol. The SPS is indifferent to the type of coherence protocol implemented at the responding node because it simply modifies its Snoop Filter to reflect the current status of the corresponding cache line as instructed by the SNC of the responding node. (Specification, page 15, line 18 – page 16, line 5; FIG. 6.)

In some instances, the responding node may be trying to send an outgoing write request at the same time it is receiving a snoop or read request from the SPS concerning the same cache line. Although the update may be included in the response to the incoming request, the home node is updated by way of an **explicit** write-back because the responding node was already in the process of updating the home node's memory location and was not implicitly generating a write-back to the home node in response to a read or snoop request. (Specification, page 16, lines 6-12.)

Figure 7 demonstrates various incoming requests that generate an implicit write-back according to one embodiment. In this embodiment of the invention, the implicit write-backs are only generated when there is no outgoing request to the same line, therefore all depicted examples assume that the cache line of the responding or dirty node is in a Modified state and that there is no outgoing write request on the cache line when the incoming request is received. (Specification, page 16, lines 13-18; FIG. 7.)

In this embodiment, five types of requests from a requesting node result in an implicit write-back by a responding node containing a copy of Modified Data

for a memory location residing on a different node than both the requesting node and the responding node. (Specification, page 16, lines 19-22.)

The Port Read Line Code/Data (PRLC/ PRLD) requests 710 are used to read a cache line. They are used to both read the data on the cache line and snoop the cache line in the caching agent at the responding node. If an IA64 coherence protocol or IOH coherence protocol is implemented at the responding node, a Snoop Response, Modified transitioning to Invalid (PSNRM) is sent to the SPS. Additionally, both the home node and the requesting node are updated with the modified data as indicated by the Route[1:0] = "11". In contrast, if the responding node is implementing the IA32 coherence protocol, a Snoop Response, Modified transitioning to Shared (PSNRS) is sent to the SPS. (Specification, page 16, line 23 – page 17, line 6; FIG. 7.)

The Port Snoop Invalidate Line/Port Snoop Invalidate Line No Data (PSIL/PSILND) requests 720 and 730 are used to snoop and invalidate a memory block at a caching node. These two request types differ in their behavior when the memory block found is in the Modified state at the snooped node. For the PSIL request, data is supplied to both the requesting node and the home node while for the PSILND request, only the home node is updated with the modified memory. For PSIL, Route[1:0] = "11" and for PSILND, Route[1:0] = "10". All three types of coherence protocols, IA64, IA32 and IOH respond the same with a PSNRM (Port Snoop Response, Modified transitioning to Invalid) response as required by the requesting node. (Specification, page 17, lines 7-15; FIG. 7.)

The Port Snoop Flush Cache Line (PSFCL) request 740 is used to flush a memory block from all the caching agents and update the home memory if the block is modified at a caching agent. This request supports flush cache instructions facilitated by various protocols such as the IA64 architectures. Again, the flush requirement invalidates all cache line states after reading the modified memory (PSNRM). Additionally, the data is updated at the home node as indicated by the Route[1:0] = "10". (Specification, page 17, lines 16-21; FIG. 7.)

In all of these cases, an implicit write-back is achieved by having the responding node initiate the update without having to wait for a separate write request to be submitted by the requesting node. (Specification, page 17, lines 22-24.)

Embodiments of the invention have been described above primarily in terms of the Assignee's Scalability Port architecture. The Implicit Write-back mechanism is not limited to use in a Distributed Shared Memory environment, nor is it limited to use in conjunction with the Assignee's Scalability Port. For instance, embodiments of the present invention may be utilized in existing or new Snoop Based architectures. (Specification, page 18, lines 2-6.)

The foregoing description has discussed the implicit write-back mechanism as being part of a hardware implemented architecture. It is understood, however, that embodiments of the invention need not be limited to such a specific application. For example, in certain embodiments the implicit

write-back mechanism could be implemented as programmable code to cooperate the activities of multiple memories located in a distributed fashion. Numerous other embodiments that are limited only by the scope and language of the claims are contemplated as would be obvious to someone possessing ordinary skill in the art and having the benefit of this disclosure. (Specification, page 18, lines 7-14.)

VI. ISSUES PRESENTED

Whether claims 1-4, and 6-14 are patentable under 35 USC §102(b) over Hamaguchi et al.

Whether claims 9-11 are patentable under 35 USC §103(a) over the combination of Hamaguchi et al. in view of Flynn et al.

Whether claims 15-18, and 20-21 are patentable under 35 USC §103(a) over Hamaguchi et al.

VII. GROUPING OF CLAIMS

For each ground of rejection contested by Appellants in this appeal, the groupings of the claims are as follows:

For purposes of the Examiner's rejection of claims 1-4 and 6-14 under 35 USC §102(b) as being anticipated by Hamaguchi et al., claims 1-4 and 6-14 stand or fall together as Group I.

For purposes of the Examiner's rejection of claims 9-11 under 35 USC §103(a) as being obvious over the combination of Hamaguchi et al. and Flynn et

al., claims 9-11 stand or fall together as Group II.

For purposes of the Examiner's rejection of claims 15-18, and 20-21 under 35 USC §103(a) as being obvious over Hamaguchi et al., claims 15-18 and 20-21 stand or fall together as Group III.

VIII. ARGUMENT

A. HAMAGUSHI ET AL. DOES NOT DISCLOSE EACH OF THE ELEMENTS RECITED IN CLAIMS 1-4 AND 6-14

As the Honorable Board is well aware, a "claim is anticipated only if *each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference*." *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987).

Nowhere does Hamaguchi teach each and every element of the Appellants' independent claims. For example, Hamaguchi does not teach the following elements:

1. "Receiving a request to read a modified cache line at a responding node."

2. "Responding to the request by updating a memory at a home node with data read from the modified cache line."

3. "Responding to the request by ... providing an answer to the requesting node."

Hamaguchi does not disclose "receiving a request to read a modified cache line at a responding node" as required by the Appellants' independent claims. In relevant part, Hamaguchi discloses that a processor (e.g., processor 22) tries to "perform a reading operation in order to ... process the data of address 1000 ..., recognizes that the data are stale... [and] generates a transaction on the bus 25 as a cache miss..." (Hamaguchi, column 4, line 66 – column 5, line 5).

The Examiner relies on this same portion of Hamaguchi to support his position. Specifically, the Examiner states "Hamaguchi clearly discloses in FIG. 3 and column 4 line 66 through column 5, line 18 that the requesting node (processor 22) tries to perform a **read** operation in order to gain access to the data of address 1000 ... (Final Office Action, Response to Arguments, page 6-7).

However, the Appellants' independent claims require "receiving a *read request* to read a modified cache line *at a responding node*". While Hamaguchi discloses performing a *read operation*, Appellants respectfully submit that a read operation is not the same as a *read request*. Hamaguchi does not disclose the concept of a read request. By the plain meaning of the terms, and as discussed in the Specification, a read request is a request to read a modified cache line that is directed to a responding node (see, for example, Specification, p. 3, lines 20-21), and that is received at a responding node. Clearly, Hamaguchi's read operation is not the same as a read request. Furthermore, even assuming that Hamaguchi does disclose a read request, Hamaguchi certainly does not disclose

a read request that is received at a responding node.

Not only is the element "receiving a read request to read a modified cache line at a responding node" missing from Hamaguchi, as discussed above, but the Examiner also fails to state that this element is disclosed by Hamaguchi. In this respect, the Examiner merely indicates that a *requesting node tries to perform a read operation*. The Examiner, however, fails to ever assert that Hamaguchi discloses "receiving a *read request* to read a modified cache line *at a responding node*" as required by the Appellants' independent claims. It is evident that the Examiner has merely read into Hamaguchi an element of Appellant's claims which is not otherwise present.

In the Advisory Action, the Examiner further argues that "receiving" and "monitoring" are analogous (see item 10). However, the Examiner fails to indicate how these terms are analogous. While the Examiner cites the phrase "receiving a snoop or read request" on page 16, line 7 of the Specification in support of his position, the Examiner does not state, and the Appellants fail to see, how this is relevant to his argument.

Nowhere in the Specification are the terms "monitoring" and "receiving" used interchangeably, and nowhere does the Specification suggest that these terms are analogous. By the usage of the terms in their associated contexts, "receiving" is not analogous to "monitoring". In Hamaguchi, a node *monitors* a bus for the presence of certain transactions. In the Appellants' invention as embodied in the claims, a responding node *receives* a read request that is issued

by a requesting node, and directed to the responding node. Also, by the plain meaning of the terms, the terms are also not analogous. For example, "monitor" may be defined as "to observe, record, or detect", and "receive" may be defined as "to take into one's possession" (Webster's Encyclopedic Unabridged dictionary of the English Language (1989)). While "receive" requires the possession of something, "monitor" merely requires the observation of that something.

Hamaguchi also does not disclose "responding to the request by updating a memory at a home node with data read from the modified cache line" as required in Appellants' independent claims. In relevant part, Hamaguchi discloses that "the processor 21 monitors the [cache miss] transaction [and] ... writes the valid data 'A" of address 1000 present in the cache memory 31 in the main storage 26" (Hamaguchi, column 5, lines 3-9). In Hamaguchi, main storage 26 is updated when a processor monitors a cache miss transaction.

In the Examiner's characterization of Hamaguchi, he states that "Hamaguchi clearly discloses in FIG. 3 and column 4 line 66 through column 5, line 18 that the responding node (processor 22) ... generates a transaction on the bus 25 as a cache miss. The cache memory control device of the responding node (processor 21) monitors the transaction and responds to the read request by placing the valid data on the bus 25 thereby, updating the main memory storage 26 and providing an answer to processor 22 (the requesting node)" (Final Office Action, Response to Arguments, Page 6 – Page 7.)

Appellants note, interestingly, that the Examiner cites the same language from Hamaguchi as do Appellants. However, the Examiner's characterization of Hamaguchi is based fully on conclusory statements. For example, the Examiner incorrectly indicates that, in Hamaguchi, a responding node responds to a read request. First, since Hamaguchi does not disclose the concept of a read request, (see discussion above), Hamaguchi certainly cannot be said to disclose responding to the read request. Furthermore, also as discussed above, main storage 26 in Hamaguchi is updated when "processor 21 monitors the transaction", and *not in response to a read request*. As disclosed in Hamaguchi, column 5, lines 4-5, this "transaction" refers to a cache miss that is generated on the bus 25. Clearly, in Hamaguchi, a processor does not place valid data on the bus in response a read request, but instead places valid data on the bus in response to monitoring a cache miss transaction.

Hamaguchi further does not disclose "responding to the request by ... providing an answer to the requesting node" as required in Appellants' independent claims. In one embodiment of the Appellants' invention as embodied by the claims, an "answer" may comprise a completion response (see, for example, Specification, page 12, lines 6-14).

In Hamaguchi, processor 21 responds to the monitoring of a cache miss transaction by "[writing] valid data 'A" of address 1000 present in the cache memory 31 in the main storage 26 and [by making] the cache memory 31 enter an SH state" (Hamaguchi, column 5, lines 8-10). Nowhere does Hamaguchi disclose "providing an answer to the requesting node".

The Examiner argues that "...[t]he cache memory control device of the responding node (processor 21) monitors the transaction and responds to the read request by placing the valid data on the bus 25 thereby, updating the main memory storage 26 and *providing an answer to processor 22* (the requesting node)" (Emphasis added, Final Office Action, Response to Arguments, Page 6 – Page 7.)

Again, while the same portions of Hamaguchi are cited by both the Appellants and the Examiner, the conclusions reached by both are very different. Again, since Hamaguchi does not disclose the concept of a read request, (see discussion above), Hamaguchi certainly cannot be said to disclose responding to the read request. Also, it appears that the Examiner attempts to imply that "[making] the cache memory 31 enter an SH state" is identical to the element "providing an answer to the requesting node". However, the Examiner has failed to describe, and the Appellants cannot find, where Hamaguchi discloses this.

Since Hamaguchi does not disclose each and every element of the Appellants' invention as embodied in the claims, the Examiner's rejection of claims 1-4 and 6-14 under 35 U.S.C. §102(b) as being anticipated by Hamaguchi is erroneous, and should be reversed.

B. THE COMBINATION OF HAMAGUCHI ET AL. AND FLYNN ET AL. DOES NOT TEACH OR SUGGEST CLAIMS 9-11 .

As the Honorable Board is well aware, in order to establish a *prima facie* case of obviousness:

First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, *the prior art reference (or references when combined) must teach or suggest all the claim limitations*." (Emphasis added). *In re Vaech*, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991). Manual of Patent Examining Procedure (MPEP), 8th Edition, August 2001, §2143.

As discussed above, nowhere does Hamaguchi teach or suggest each and every element of the Appellants' independent claims. For example, Hamaguchi does not teach the following elements:

1. "Receiving a request to read a modified cache line at a responding node."

2. "Responding to the request by updating a memory at a home node with data read from the modified cache line."

3. "Responding to the request by ... providing an answer to the requesting node."

With respect to claims 9-11, which depend from claim 8, the Examiner cites Flynn for teaching the usage of a centralized system control unit to maintain a copy of a cache directory as well as the presence vector of each processor caches. However, since Hamaguchi fails to disclose many of the elements required by the Appellants' independent claims, including claim 8, and since Flynn fails to disclose, teach and/or suggest those elements missing from Hamaguchi, the combination of Hamaguchi and Flynn fails to teach or suggest each and every element of the Appellants' invention as embodied in the claims. Consequently, the Examiner has not established a prima facie case of obviousness, and the Examiner's rejection of claims 9-11 under 35 U.S.C. §103(a) as being obvious over the combination of Hamaguchi and Flynn should be reversed.

C. HAMAGUCHI ET AL. DOES NOT TEACH OR SUGGEST CLAIMS 15-18 AND 20-21.

Again, as discussed above, nowhere does Hamaguchi teach or suggest each and every element of the Appellants' independent claims. For example, Hamaguchi does not teach the following elements:

1. "Receiving a request to read a modified cache line at a responding node."

2. "Responding to the request by updating a memory at a home node with data read from the modified cache line."

3. "Responding to the request by ... providing an answer to the

requesting node."

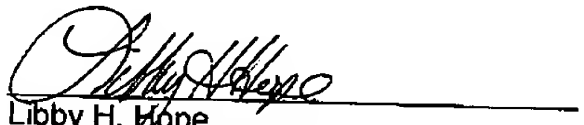
Claims 15-18 and 20-21 recite elements similar to those recited in claims 1-4 and 6-14. With respect to claims 15-18 and 20-21, the Examiner cites Hamaguchi for teaching the usage of a computer readable medium to accomplish the operations recited in the claims. However, Hamaguchi still does not disclose, teach, or otherwise suggest each and every element of the claims, including the operations recited in the claims, as discussed above. Therefore, since Hamaguchi does not teach or suggest each and every element of claims 15-18 and 20-21, the Examiner has not established a prima facie case of obviousness, and the Examiner's rejection of claims 15-18 and 20-21 under 35 U.S.C. §103(a) as being obvious over Hamaguchi should be reversed.

VII. CONCLUSION

For the reasons discussed above, Appellants respectfully submit that each and every one of the final rejections made by the Examiner in the Final Office Action is erroneous. Accordingly, Appellants respectfully request that the Honorable Board of Patent Appeals and Interferences reverse the Examiner and direct that all of the currently pending claims be allowed.

Respectfully submitted,

Date: June 15, 2004



Libby H. Hope
Attorney for Appellants
Reg. No. 46,774
Patent Practice Group
INTEL CORPORATION

APPENDIX A

1. A method comprising:

receiving a request to read a modified cache line at a responding node of a shared memory multiprocessor architecture from a requesting node of the shared memory multiprocessor architecture; and

responding to the request by updating a memory at a home node with data read from the modified cache line, and by providing an answer to the requesting node, wherein the home node is different from the responding node.

2. The method of claim 1, wherein the answer includes a copy of the data read from the modified cache line.

3. The method of claim 1, wherein the response further provides a status of the modified cache line.

4. The method of claim 3, wherein the status indicates one of:

the modified cache line is transitioning from a modified state to an invalid state; and

the modified cache line is transitioning from a modified state to a shared state.

6. The method of claim 1, further comprising providing a completion response to the requesting node.

7. The method of claim 3, wherein the status indicates a cache coherence protocol type used by the responding node.

8. A shared memory multiprocessor system comprising:

a plurality of node controllers and a switch coupled to each of the plurality of node controllers configured to:

transmit a read request regarding a modified cache line from a first node controller of the plurality of node controllers through the switch to a second node controller of the plurality of node controllers, wherein the second node controller is distinct from the first node controller; and

in response to receiving the read request regarding the modified cache line, the second node controller instructs the switch to update a home memory residing exclusively on a third node controller of the plurality of node controllers.

9. The shared memory multiprocessor system of claim 8 wherein the switch maintains a presence vector.

10. The shared memory multiprocessor system of claim 9 wherein the presence vector maintains a status of a cache line for each participating node controller of the plurality of node controllers.

11. The shared memory multiprocessor system of claim 10 wherein the presence vector indicates if the cache line for each corresponding

participating node controller contains a copy of a contents stored in the home memory.

12. A method comprising a responding node initiating an implicit write-back in response to a read request directed to a modified cache line at the responding node.

13. The method of claim 12, wherein the implicit write-back includes information causing the read request to be answered and a home memory to be updated.

14. The method of claim 12, wherein the implicit write-back further includes information identifying a state of the modified cache line.

15. A machine-readable medium having stored thereon data representing sequences of instructions, the sequences of instructions which, when executed by a processor, cause the processor to:

receive a request to read a cache line at a responding node of a shared memory multiprocessor architecture from a requesting node of the shared memory multiprocessor architecture; and

respond to the request by updating a memory at a home node with data read from the cache line, and by providing an answer to the requesting node, wherein the home node is different from the responding node.

16. The machine-readable medium of claim 15 wherein the answer includes a

copy of the data read from the cache line.

17. The machine-readable medium of claim 15, wherein the response further provides a status of the cache line.

18. The machine-readable medium of claim 17, wherein the status indicates one of:

the cache line is transitioning from a modified state to an invalid state; and

the cache line is transitioning from a modified state to a shared state.

20. The machine-readable medium of claim 15, wherein the sequence of instructions further causes the processor to provide a completion response to the requesting node.

21. The machine-readable medium of claim 17, wherein the status indicates a cache coherence protocol type used by the responding node.